Лабораторная работа №11
Структуры

Контрольные вопросы:

1) Как объявляются новые типы (псевдонимы) и для чего это нужно?
2) Что такое структура, для чего она нужна?
3) Что такое поля структуры и как они размещаются в памяти?
4) Как происходит объявление структуры?
5) Как происходит копирование структуры?
6) Как использовать указатель на структуру и как обращаться к полям указателя на структуру?
7) Какими способами можно передавать структуру в качестве аргумента функции и в чём отличие между ними?

Задание

В данной лабораторной работе необходимо создать структуру данных, представляющую собой некоторую сущность: вектор, дробь, фигура и т.д. (по вариантам ниже), а также 8 операций (функций) над этой структурой. Программа должна запрашивать у пользователя какую операцию необходимо протестировать, далее запрашивать данные для операндов и показывать результат операции с помощью функции «вывод в консоль». В качестве упрощения, задание не подразумевает использование динамической памяти, следовательно, во всех вариантах используются статические массивы (если это обусловлено задачей).

Важно: структуры создаются с помощью отдельной функции, возвращающей структуру по значению. Это необходимо, например, для того чтобы рассчитать некоторые поля структуры на основе переданных данных. Пример:

```
typedef struct
{
    char name[30];
    unsigned short yearBirth;
    unsigned char age; // поле, которое будет рассчитано при создании
} Person;


Person createPerson(const char* name, unsigned short yearBirth)
{
    Person person;
    strcpy(person.name,name);
    person.yearBirth = yearBirth;
    person.age = getCurrentYear() - yearBirth; //рассчитываем поле age
    return person; //возвращаем структуру по значению
}
int main()
{
    Person person = createPerson("Ivan Ivanov", 1991);

    printf("name: %s year of birth: %d age: %d", person.name,
                                                 person.yearBirth,
                                                 person.age);

    return 0;
}
```

В данном примере поле age было рассчитано автоматически при создании структуры с помощью функции createPerson. Это весьма удобно и в том случае, если внутри структуры данные представлены другим способом, нежели нам удобно их задавать при инициализации.
Также возвращение по значению полезно, когда нам не нужно модифицировать исходную структуру, а нужно создать отдельную новую, которая будет хранить результат операции. В следующем примере мы создаём структуру для представления точки с целочисленными координатами и функцию для её относительного сдвига:

```
typedef struct
{
    int x;
    int y;
} Point;


Point moveTo(const Point* point, int dx, int dy)
{
    Point newPoint = {point->x + dx, point->y + dy};
    return newPoint; //возвращаем структуру по значению
}

int main()
{
    Point point1 = {0, 5};
    Point point2 = moveTo(&point1, 1, -3);
    Point point3 = moveTo(&point2, 4, 0);

    printf("x1:%d y1:%d\n"
           "x2:%d y2:%d\n"
           "x3:%d y3:%d\n",
           point1.x, point1.y,
           point2.x, point2.y,
           point3.x, point3.y);

    //  x1:0 y1:5
    //  x2:1 y2:2
    //  x3:5 y3:2

    return 0;
}
```

В данном примере в функции main мы дважды смещаем точку, и каждый раз результат возвращается временным объектом который мы сразу копируем в новую переменную, и после этого все три координаты точек у нас сохранены и готовы для дальнейшего использования.

Может показаться, что передача по значению локальной переменной-структуры из функции крайне неэффективно, т.к. происходит несколько раз объявление и копирование переменной. Однако в реальности не всё так плохо: в современных компиляторах присутствует **NRVO**-оптимизация (**Named Return Value Optimization**). Она заключается в том, что компилятор видя, что локальная переменная используется в качестве временного объекта для возвращаемого значения, выбросит создание этой переменной и будет работать уже непосредственно с той переменной, в результате к которой присваивается значение функции, что существенно повышает производительность. Без этой оптимизации была бы сначала создана локальная переменная-структура и

проинициализирована значениями, затем при возврате значения функцией была бы создана копия данной структуры как временный объект, который затем опять был бы опять скопирован в созданную в основной программе переменную-структуру.

Важно: во всех функциях где написано «*результат возвращается как новое значение*» следует создавать и возвращать новую структуру, во всех других случаях следует модифицировать существующую, которая была передана в качестве параметра.

Варианты заданий

Вариант 1. «**Комплексное число**» – **Complex**.
Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). Должны быть следующие функции:
1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.
2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.
3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.
4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.
5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.
6) Преобразование переданного комплексного числа в сопряжённое.
7) Преобразование переданного комплексного числа в обратное.
8) Вывод в консоль переданного комплексного числа в виде:
`(a + b * i) [real],`
где real имеет значение либо «*комплексное*», либо «*вещественное*».

Вариант 2. «**Дробь**» – **Fraction.**
Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m,n*) а также отдельно логического значения, указывающего на знак дроби. Должны быть следующие функции:
1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.
2) Сложение двух дробей, как результат возвращается новая дробь.
3) Вычитание двух дробей, как результат возвращается новая дробь.
4) Умножение двух дробей, как результат возвращается новая дробь.
5) Деление двух дробей, как результат возвращается новая дробь.
6) Приведение двух переданных дробей к общему знаменателю.
7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».
8) Вывод в консоль переданной дроби в виде:
`[sign] a / b,`
где sign имеет значение либо «*+*», либо «-».

Вариант 3. «**Двумерный вектор**» – **Vector2d.**
Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. Должны быть следующие функции:
1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Умножение переданного вектора на заданный скаляр.
5) Нормализация переданного вектора.
6) Получение значения скалярного произведения двух векторов.
7) Установка заданной длины для переданного вектора.
8) Вывод в консоль переданного вектора в виде:
`(x,y) [norm],`
где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 4. «**Трёхмерный вектор**» – **Vector3d.**
Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. Должны быть следующие функции:
1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Получение значения угла между двумя переданными векторами.
5) Нормализация переданного вектора.
6) Векторное произведение двух векторов, как результат возвращается новый вектор.
7) Проекция вектора на вектор, как результат возвращается новый вектор.
8) Вывод в консоль переданного вектора в виде:
`(x,y,z) [norm],`
где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 5. «**Квадратная матрица 3x3**» – **Matrix3x3**.
Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. Должны быть следующие функции:
1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
2) Сложение двух матриц, как результат возвращается новая матрица.
3) Умножение двух матриц, как результат возвращается новая матрица.
4) Умножение переданной матрицы на скаляр.
5) Преобразование переданной матрицы в транспонированную.
6) Нахождение определителя переданной матрицы.
7) Преобразование переданной матрицы в обратную.
8) Вывод в консоль переданной матрицы в виде:

```
-1.4        3        2.5
 3.2        1       -0.4
```

```
 3.2     4.3         4
[identity]
```
где identity имеет значение либо «единичная», либо «не единичная»

Вариант 6. «**Квадратная матрица 4x4**» – **Matrix4x4**.
Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. Должны быть следующие функции:
1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
2) Сложение двух матриц, как результат возвращается новая матрица.
3) Умножение двух матриц, как результат возвращается новая матрица.
4) Умножение переданной матрицы на скаляр.
5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.
6) Нахождение определителя переданной матрицы.
7) Преобразование переданной матрицы в обратную.
8) Вывод в консоль переданной матрицы в виде:
```
-1.4       3       2.5     3.2
 3.2       1      -0.4      -3
 3.2     4.3         4     1.2
  -4     3.1      -3.4       2
[zero]
```
где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 7. «Прямоугольник» – Rectangle.
Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. Должны быть следующие функции:
1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.
2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.
3) Изменение ширины и высоты для переданного прямоугольника.
4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.
5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.
6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.
7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.
8) Вывод в консоль переданного прямоугольника в виде:
```
(0, 5)   (10.5, 5)
(0,0)    (10.5,0)
[square]
```
где square имеет значение либо «квадрат», либо «прямоугольник».

Вариант 8. «Круг» – Circle.

Разработать структуру данных Circle, представляющую фигуру круг, заданный координатами его центра и радиусом. Должны быть следующие функции:
1) Создание структуры круга по координатам его центра и радиусу. Как результат возвращается новый круг.
2) Определение попадания заданной точки в переданный круг.
3) Определение факта пересечения двух переданных кругов.
4) Определение факта пересечения переданного круга с прямоугольной областью с параметрами (x,y,width,height).
5) Рассчитать площадь пересечения двух переданных кругов.
6) Рассчитать круг, описанный вокруг треугольника, заданный тремя координатами вершин. Как результат возвращается новый круг.
7) Рассчитать круг минимального радиуса, внутрь которого попадают все круги, переданные в качестве массива. Как результат возвращается новый круг.
8) Вывод в консоль переданного прямоугольника в виде:

```
(x,y) [R]
```

Вариант 9. «Треугольник» - Triangle.
Разработать структуру данных Triangle, представляющую фигуру треугольник, заданный координатами его вершин, а также логическим значением equilateral, показывающим является ли данный треугольник равносторонним. Должны быть следующие функции:
1) Создание структуры треугольника по координатам его вершин. Как результат возвращается новый треугольник.
2) Перемещение переданного треугольника на заданное смещение относительно осей X и Y.
3) Расчёт координат центроида переданного треугольника.
4) Поворот переданного треугольника на заданный угол вокруг его центроида.
5) Изменение размера переданного треугольника на заданный коэффициент масштаба относительно его центроида.
6) Создание равностороннего треугольника по заданному размеру стороны, центроид которого совпадает с точкой (0,0). Как результат возвращается новый треугольник.

7) Создание прямоугольного треугольника по двум заданным длинам катетов, совпадающих с положительными полуосями X и Y. Как результат возвращается новый треугольник.
8) Вывод в консоль переданного треугольника в виде:

```
            (3, 9.5)
(2.3, 3.4)          (8.6, 2.5)
[equilateral]
```

где equilateral имеет значение либо «равносторонний», либо «не равносторонний».

Вариант 10. «Многочлен» – Polynom.
Разработать структуру данных **Polynom**, представляющую многочлен от одной переменной вида $ax^n+bx^{n-1}+…dx + e$ степени n (максимальная степень 10), заданный статическим массивом коэффициентов (e, d … b, a) и без знаковым целым числом, выражающее степень n. Должны быть следующие функции:
1) Создание структуры многочлена по переданному массиву коэффициентов. Как результат возвращается новый многочлен.
2) Сложение двух многочленов, как результат возвращается новый многочлен.
3) Вычитание двух многочленов, как результат возвращается новый многочлен.

4) Умножение двух многочленов, как результат возвращается новый многочлен.
5) Деление двух многочленов, как результат возвращается новый многочлен (остаток отбрасывается).
6) Умножение переданного многочлена на число.
7) Расчёт значения переданного многочлена для заданного значения переменной.
8) Вывод в консоль переданного многочлена в виде:
`a * x^n + b * x^n-1 + … + d * x + e`

Вариант 11. «**Множество целых чисел**» – **Set.**
Разработать структуру данных Set, представляющую множество целочисленных элементов (с максимальным количеством элементов 50), представленной статическим массивом элементов, а также целочисленной переменной size, задающее текущий размер множества. Должны быть следующие функции:
1) Создание структуры множества по переданному массиву элементов. Как результат возвращается новое множество.
2) Принадлежность заданного элемента переданному множеству.
3) Добавление элемента к переданному множеству.
4) Удаление элемента из переданного множества.
5) Создать множество, являющееся пересечением двух переданных множеств. Как результат возвращается новое множество.
6) Создать множество, являющееся объединением двух переданных множеств. Как результат возвращается новое множество.
7) Создать множество, являющееся вычитанием двух переданных множеств. Как результат возвращается новое множество.
8) Вывод в консоль переданного множества в виде:
`(a,b,c,d,e…) [size]`
где size – размер множества.

Вариант 12. «**Комплексное число**» – **Complex**.
Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). Должны быть следующие функции:
1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.
2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.
3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.
4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.
5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.
6) Преобразование переданного комплексного числа в сопряжённое.
7) Преобразование переданного комплексного числа в обратное.
8) Вывод в консоль переданного комплексного числа в виде:
`(a + b * i) [real],`
где real имеет значение либо «*комплексное*», либо «*вещественное*».

Вариант 13. «**Дробь**» – **Fraction.**
Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m,n*) а также отдельно логического значения, указывающего на знак дроби. Должны быть следующие функции:
1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.
2) Сложение двух дробей, как результат возвращается новая дробь.
3) Вычитание двух дробей, как результат возвращается новая дробь.
4) Умножение двух дробей, как результат возвращается новая дробь.
5) Деление двух дробей, как результат возвращается новая дробь.
6) Приведение двух переданных дробей к общему знаменателю.
7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».
8) Вывод в консоль переданной дроби в виде:
`[sign] a / b,`
где sign имеет значение либо «*+*», либо «-».

Вариант 14. «**Двумерный вектор**» – **Vector2d.**
Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. Должны быть следующие функции:
1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Умножение переданного вектора на заданный скаляр.
5) Нормализация переданного вектора.
6) Получение значения скалярного произведения двух векторов.
7) Установка заданной длины для переданного вектора.
8) Вывод в консоль переданного вектора в виде:
`(x,y) [norm],`
где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 15. «**Трёхмерный вектор**» – **Vector3d.**
Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. Должны быть следующие функции:
1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Получение значения угла между двумя переданными векторами.
5) Нормализация переданного вектора.
6) Векторное произведение двух векторов, как результат возвращается новый вектор.
7) Проекция вектора на вектор, как результат возвращается новый вектор.
8) Вывод в консоль переданного вектора в виде:
`(x,y,z) [norm],`
где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 16. «**Квадратная матрица 3x3**» – **Matrix3x3**.
Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. Должны быть следующие функции:
1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
2) Сложение двух матриц, как результат возвращается новая матрица.
3) Умножение двух матриц, как результат возвращается новая матрица.
4) Умножение переданной матрицы на скаляр.
5) Преобразование переданной матрицы в транспонированную.
6) Нахождение определителя переданной матрицы.
7) Преобразование переданной матрицы в обратную.
8) Вывод в консоль переданной матрицы в виде:

```
-1.4        3        2.5
 3.2        1       -0.4
 3.2      4.3          4
[identity]
```

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 17. «**Квадратная матрица 4x4**» – **Matrix4x4**.
Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. Должны быть следующие функции:
1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
2) Сложение двух матриц, как результат возвращается новая матрица.
3) Умножение двух матриц, как результат возвращается новая матрица.
4) Умножение переданной матрицы на скаляр.
5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.
6) Нахождение определителя переданной матрицы.
7) Преобразование переданной матрицы в обратную.
8) Вывод в консоль переданной матрицы в виде:

```
-1.4        3        2.5      3.2
 3.2        1       -0.4       -3
 3.2      4.3          4      1.2
  -4      3.1       -3.4        2
[zero]
```

где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 18. «Прямоугольник» – Rectangle.
Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. Должны быть следующие функции:
1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.
2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.
3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.
5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.
6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.
7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.
8) Вывод в консоль переданного прямоугольника в виде:

```
(0, 5)   (10.5, 5)
(0,0)    (10.5,0)
[square]
```

где square имеет значение либо «квадрат», либо «прямоугольник».

Вариант 19. «Круг» – Circle.

Разработать структуру данных Circle, представляющую фигуру круг, заданный координатами его центра и радиусом. Должны быть следующие функции:
1) Создание структуры круга по координатам его центра и радиусу. Как результат возвращается новый круг.
2) Определение попадания заданной точки в переданный круг.
3) Определение факта пересечения двух переданных кругов.
4) Определение факта пересечения переданного круга с прямоугольной областью с параметрами (x,y,width,height).
5) Рассчитать площадь пересечения двух переданных кругов.
6) Рассчитать круг, описанный вокруг треугольника, заданный тремя координатами вершин. Как результат возвращается новый круг.
7) Рассчитать круг минимального радиуса, внутрь которого попадают все круги, переданные в качестве массива. Как результат возвращается новый круг.
8) Вывод в консоль переданного прямоугольника в виде:

```
(x,y) [R]
```

Вариант 20. «Треугольник» - Triangle.

Разработать структуру данных Triangle, представляющую фигуру треугольник, заданный координатами его вершин, а также логическим значением equilateral, показывающим является ли данный треугольник равносторонним. Должны быть следующие функции:
1) Создание структуры треугольника по координатам его вершин. Как результат возвращается новый треугольник.
2) Перемещение переданного треугольника на заданное смещение относительно осей X и Y.
3) Расчёт координат центроида переданного треугольника.
4) Поворот переданного треугольника на заданный угол вокруг его центроида.
5) Изменение размера переданного треугольника на заданный коэффициент масштаба относительно его центроида.
6) Создание равностороннего треугольника по заданному размеру стороны, центроид которого совпадает с точкой (0,0). Как результат возвращается новый треугольник.

7) Создание прямоугольного треугольника по двум заданным длинам катетов, совпадающих с положительными полуосями X и Y. Как результат возвращается новый треугольник.

8) Вывод в консоль переданного треугольника в виде:

```
            (3, 9.5)
(2.3, 3.4)          (8.6, 2.5)
[equilateral]
```

где equilateral имеет значение либо «равносторонний», либо «не равносторонний».

Вариант 21. «Многочлен» – Polynom.

Разработать структуру данных **Polynom**, представляющую многочлен от одной переменной вида $ax^n+bx^{n-1}+…dx + e$ степени n (максимальная степень 10), заданный статическим массивом коэффициентов (e, d … b, a) и без знаковым целым числом, выражающее степень n. Должны быть следующие функции:

1) Создание структуры многочлена по переданному массиву коэффициентов. Как результат возвращается новый многочлен.
2) Сложение двух многочленов, как результат возвращается новый многочлен.
3) Вычитание двух многочленов, как результат возвращается новый многочлен.
4) Умножение двух многочленов, как результат возвращается новый многочлен.
5) Деление двух многочленов, как результат возвращается новый многочлен (остаток отбрасывается).
6) Умножение переданного многочлена на число.
7) Расчёт значения переданного многочлена для заданного значения переменной.
8) Вывод в консоль переданного многочлена в виде:

```
a * x^n + b * x^n-1 + … + d * x + e
```

Вариант 22. «**Множество целых чисел**» – **Set.**

Разработать структуру данных Set, представляющую множество целочисленных элементов (с максимальным количеством элементов 50), представленной статическим массивом элементов, а также целочисленной переменной size, задающее текущий размер множества. Должны быть следующие функции:

1) Создание структуры множества по переданному массиву элементов. Как результат возвращается новое множество.
2) Принадлежность заданного элемента переданному множеству.
3) Добавление элемента к переданному множеству.
4) Удаление элемента из переданного множества.
5) Создать множество, являющееся пересечением двух переданных множеств. Как результат возвращается новое множество.
6) Создать множество, являющееся объединением двух переданных множеств. Как результат возвращается новое множество.
7) Создать множество, являющееся вычитанием двух переданных множеств. Как результат возвращается новое множество.
8) Вывод в консоль переданного множества в виде:

```
(a,b,c,d,e…) [size]
```

где size – размер множества.